Parallel programming using OpenMP-2

Xuenan Cui, xncui@inha.ac.kr

Computer Vision Lab.

School of Information & Communication Engineering

Inha University

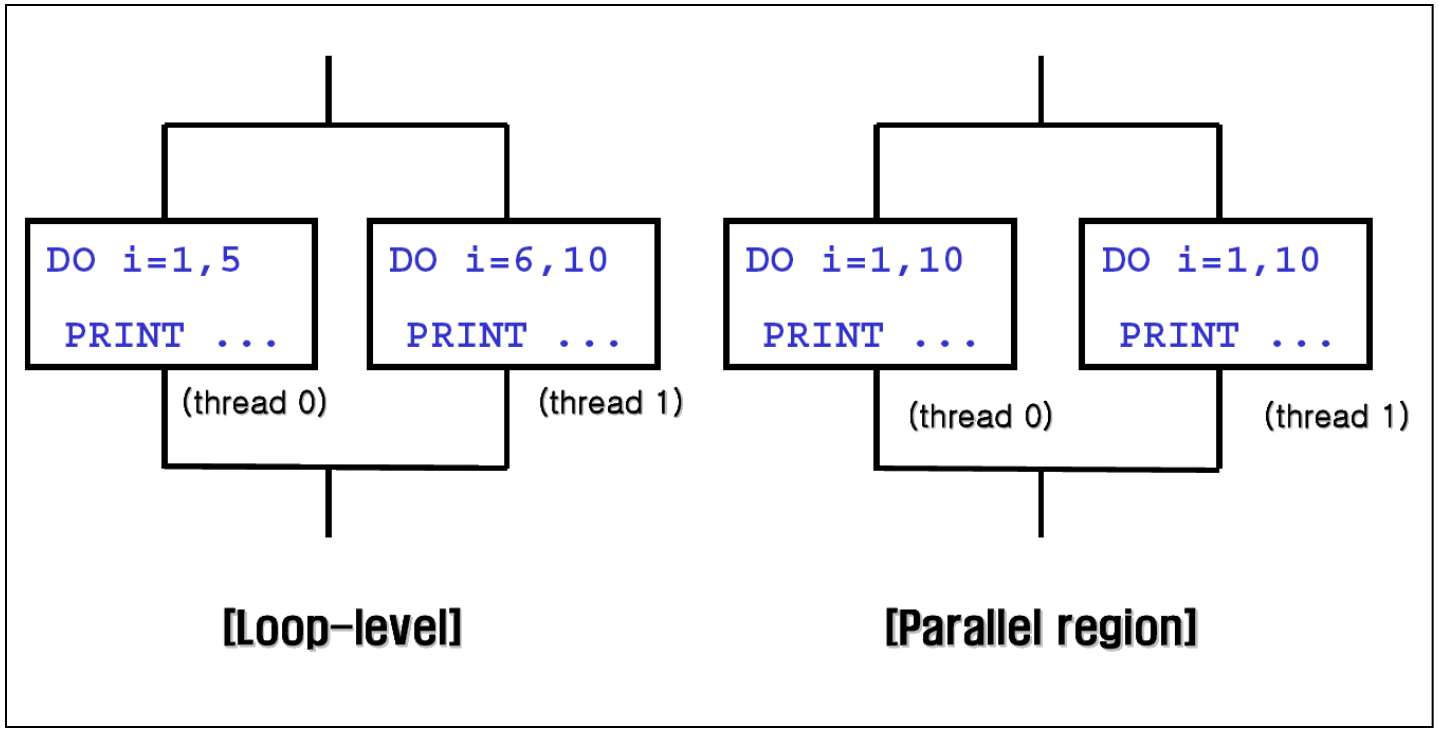
Parallel programming using OpenMP

Consideration of the parallel programming

- Considerations
  - The processing time must be faster than serial
  - The results are same as serial

Loop level & parallel region

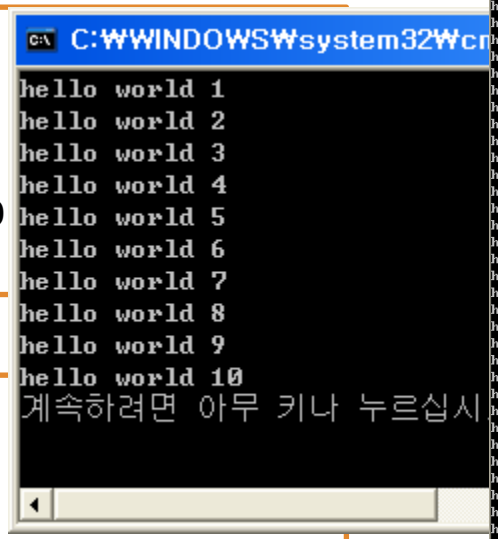
- Compare to the loop level & parallel region parallelization

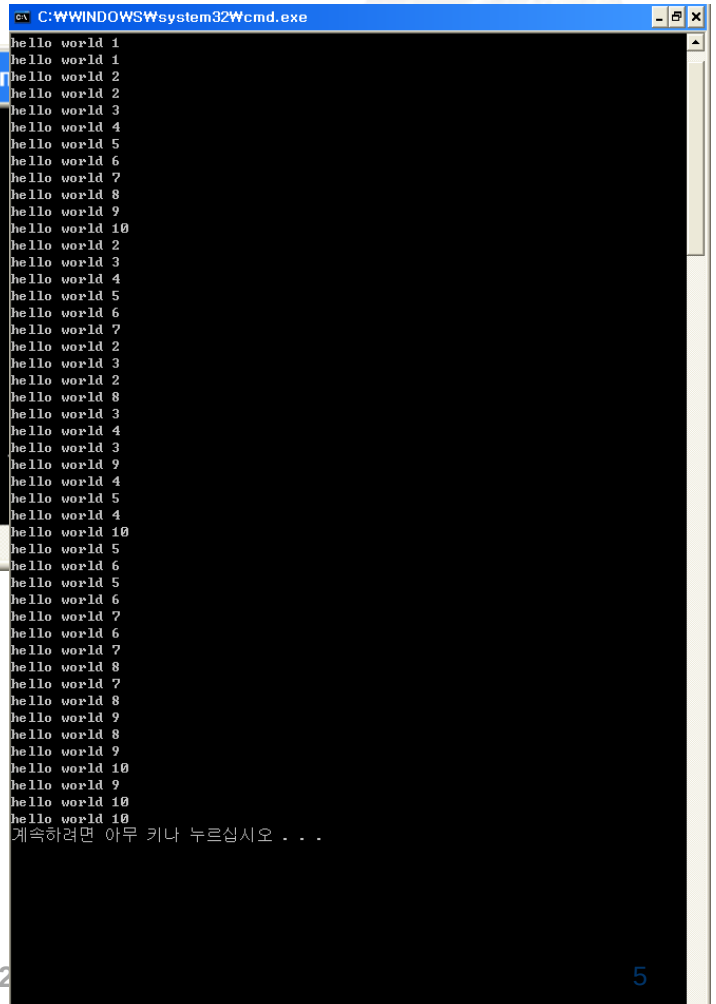


Loop level & parallel region

```
#pragma omp parallel for
        for(i=1;i<=10;i++)
                printf("hello wo
```

```
#pragma omp parallel
        for(i=1;i<=10;i++)
                printf("hello world %d\n",i);
```





Constraints of parallel syntax usage

- *Parallel for* only uses in the for loop
  - Disabled to the While, do-while
- Disabled to *return, goto, break* in the parallel region
- *Parallel /end parallel* can include other *Parallel /end parallel* ,but second part execute in serial
- *Parallel for* can not include other *parallel for*

Constraints of parallel syntax usage

```
#pragma omp for
        for(int i=1;i<=10;i++){
                printf("hello world %d\n",i);
#pragma omp for
        for(int i=1;i<=10;i++){
                printf("hello world %d\n",i);
        }
        }
```

```
1>------ 모두 다시 빌드 시작: 프로젝트: test, 구성: Debug Win32 ------
1>'test' 프로젝트, 'Debug|Win32' 구성에 사용할 중간 파일 및 출력 파일을 삭제하고 있습니다...
1>컴파일하고 있습니다...
1>tomp.cpp
1>c:\documents and settings\최학남\my documents\visual studio 2005\projects\test\test\tomp.cpp(28) : error C3034: OpenMP 'for' 지시문은 'for' 지시문 내부에 직접 중첩될 수 없습니다.
1>빌드 로그가 "file://c:\Documents and Settings\최학남\My Documents\Visual Studio 2005\Projects\test\test\Debug\BuildLog.htm"에 저장되었습니다.
1>test - 오류: 1개, 경고: 0개
========== 모두 다시 빌드: 성공 0, 실패 1, 생략 0 ==========
```

Synchronization

- Data dependence

```
a={1,2,3,4,5};
For (i=1;i<5;i++)
    a(i)=a(i)*a(i-1);
```

⇨ Output : a(5)=120

Thread 0	Thread 1
a(2)=a(2)*a(1)=2*1=2	
	a(4)=a(4)*a(3)=4*3=12
a(3)=a(3)*a(2)=3*2=6	
	a(5)=a(5)*a(4)=5*12=110

Synchronization

- Data dependence

```
int a[5]={1,2,3,4,5};
for (int i=1;i<5;i++)
  {
     a[i]=a[i]*a[i-1];
     printf("%d\n",a[i]);
  }
```

```
C:₩WINDOWS₩system32₩cmd.exe
2
6
24
120
계속하려면 아무 키나 누르십시오 . . .
```

```
int a[5]={1,2,3,4,5};
#pragma omp parallel  for
for (int i=1;i<5;i++)
  {
     a[i]=a[i]*a[i-1];
     printf("%d\n",a[i]);
  }
```

```
C:₩WINDOWS₩system32₩cmd.exe
a is 12
a is 2
a is 6
a is 60
계속하려면 아무 키나 누르십시오 . . .
```

Synchronization

- Master directive
  - #pragma omp master
  - Execute in the master thread
  - Salver thread skip to the master part

Synchronization

❖ Master directive

```
int a[10]={1,6,4,2,7,3,10,5,9,8};
int b[10]={10,1,5,9,8,6,4,2,7,3};
int c[10]={2,7,3,10,1,6,4,5,9,8};
#pragma omp parallel private(Myid)
{
  #pragma omp for
  for(int i=0;i<10;i++)
  {
     a[i] = b[i]+c[i];
     printf("a[i]is %d.  thread ID is %d\n",a[i],omp_get_thread_num());
  }
  printf("++++++++++++++++++++++++++++\n");
  #pragma omp master
  {
  for(int j=0;j<10;j++)
    {
       b[j] = a[j]-c[j];
      printf("b[i]is %d.  master thread is %d\n", b[j], omp_get_thread_num());
    }
  }
}
```

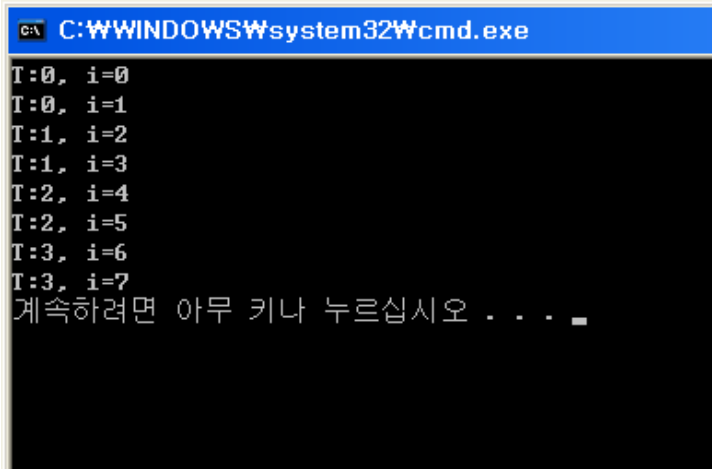
```
C:₩WINDOWS₩system32₩cmd.exe
a[i]is 12.  thread ID is 1
a[i]is 12.  thread ID is 0
a[i]is 8.  thread ID is 1
a[i]is 8.  thread ID is 0
a[i]is 7.  thread ID is 1
a[i]is 8.  thread ID is 0
a[i]is 16.  thread ID is 1
a[i]is 19.  thread ID is 0
a[i]is 11.  thread ID is 1
a[i]is 9.  thread ID is 0
++++++++++++++++++++++++++++
++++++++++++++++++++++++++++
b[i]is 10.  master thread is 0
b[i]is 1.  master thread is 0
b[i]is 5.  master thread is 0
b[i]is 9.  master thread is 0
b[i]is 8.  master thread is 0
b[i]is 6.  master thread is 0
b[i]is 4.  master thread is 0
b[i]is 2.  master thread is 0
b[i]is 7.  master thread is 0
b[i]is 3.  master thread is 0
계속하려면 아무 키나 누르십시오 . . .
```

Synchronization

- ordered directive
  - #pragma omp ordered
    - In the *ordered* region execute to serially
    - Only one thread is allowed in an ordered section at any time
    - Only one *ordered* region allowed In the *parallel for* region

Synchronization

```
int Myid,i;
omp_set_num_threads(4);
#pragma omp parallel private(Myid)
{
      Myid = omp_get_thread_num();
      #pragma omp for private(i) ordered
      for(i=0;i<8;i++)
            {
               #pragma omp ordered
               printf("T:%d, i=%d\n",Myid, i);
            }
}
```



Exercise 1:Image Filtering using OpenMP parallel for

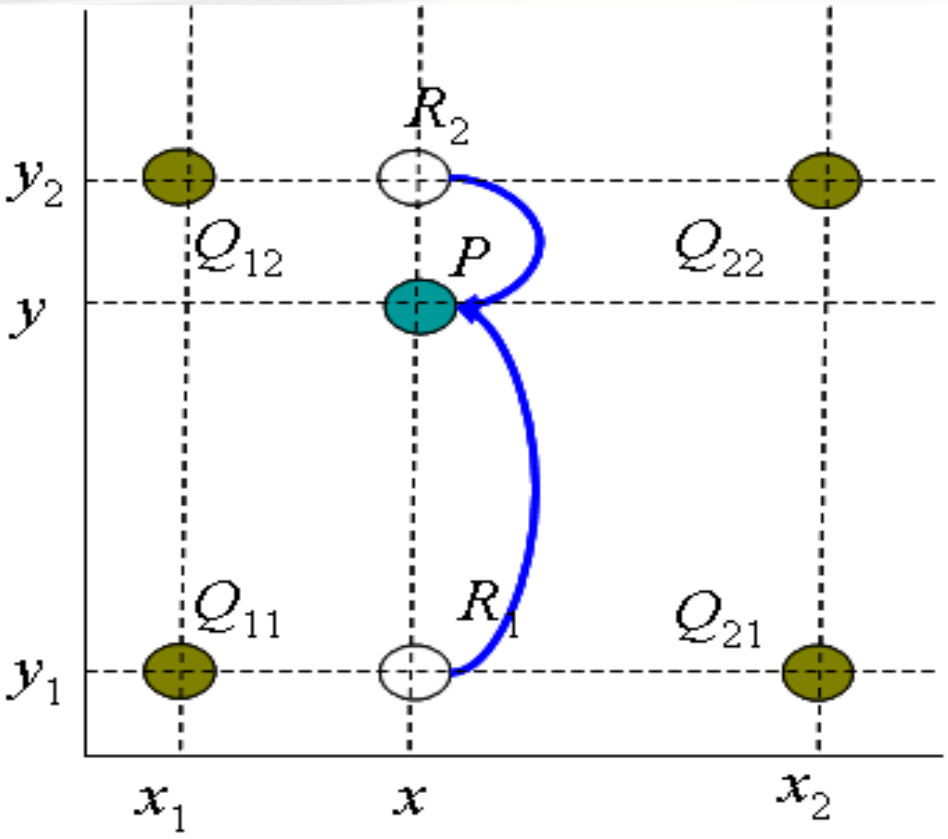
- Load image using OpenCV
- *Filter2DCV(src,h,w, dst, Elements, hE,wE)*
- Apply the OpenMP to the serial code
  - *Parallel omp for* and *parallel omp sections*

- **Compare to the processing time between serial and two OpenMP results**

HW#3-Interpolation using OpenMP

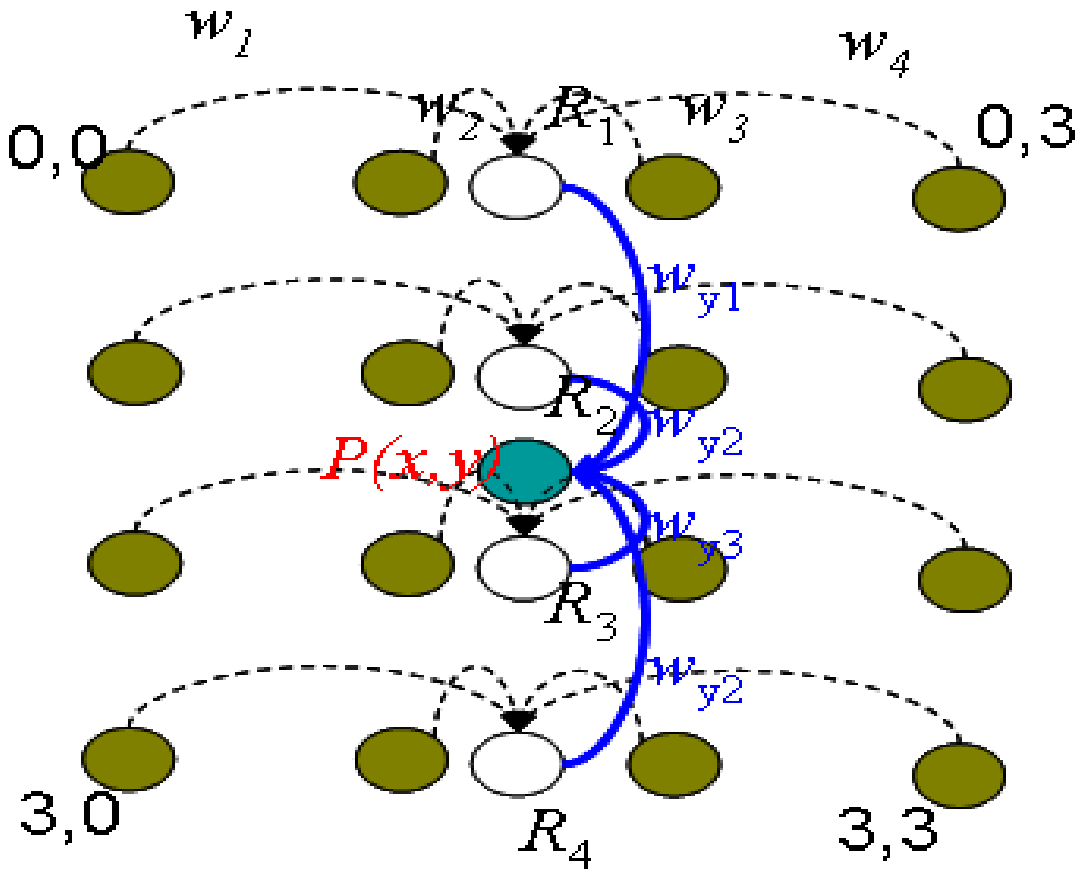
- Introduction to interpolation methods
- Programming steps
- Style of reporting

(1) Bilinear interpolation



$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

(2) Bi-cubic interpolation



(2) Bi-cubic interpolation

a = -0.5

$$w_{cubic\_4} = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \le |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \le |x| < 2 \\ 0 & elsewhere \end{cases}$$

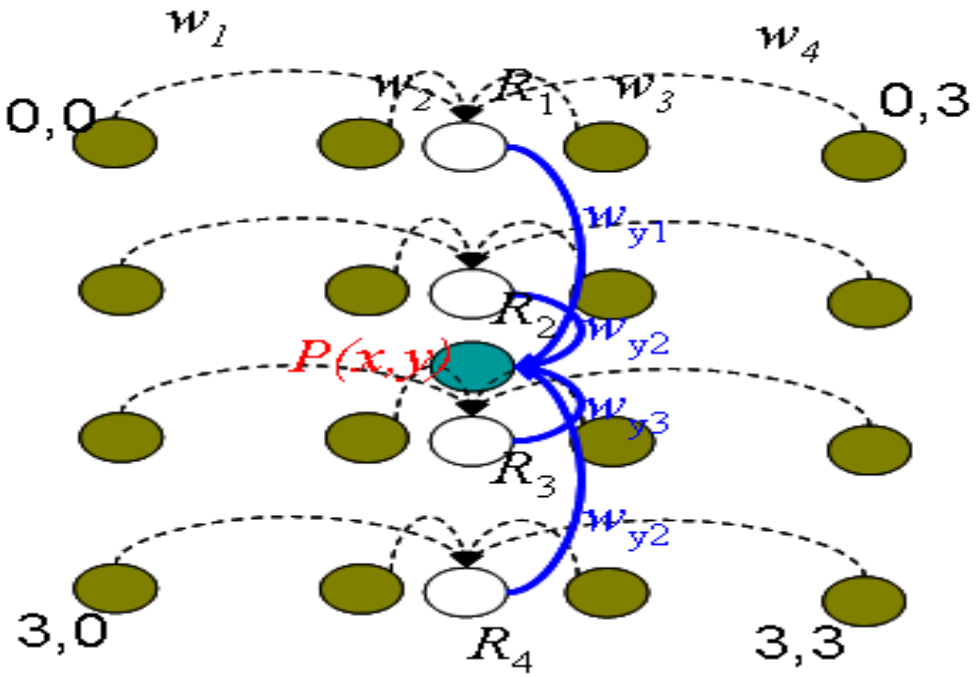
$$w_{cubic\_6} = \begin{cases} (6/5)|x|^3 - (11/5)|x|^2 + 1 & 0 \le |x| < 1 \\ -(3/5)|x|^3 + (16/5)|x|^2 - (27/5)|x| + 14/5 & 1 \le |x| < 2 \\ (1/5)|x|^3 - (8/5)|x|^2 + (21/5)|x| - 18/5 & 2 \le |x| < 3 \\ 0 & elsewhere \end{cases}$$

(2) Bi-cubic interpolation

$$w_{cubic\_8} = \begin{cases} (67/56)|x|^3 - (123/56)|x|^2 + 1 & 0 \le |x| < 1 \\ -(33/56)|x|^3 + (177/56)|x|^2 - (75/14)|x| + 39/14 & 1 \le |x| < 2 \\ (9/56)|x|^3 - (75/56)|x|^2 + (51/14)|x| - 45/14 & 2 \le |x| < 3 \\ -(3/56)|x|^3 + (33/56)|x|^2 - (15/7)|x| + 18/7 & 3 \le |x| < 4 \\ 0 & elsewhere \end{cases}$$

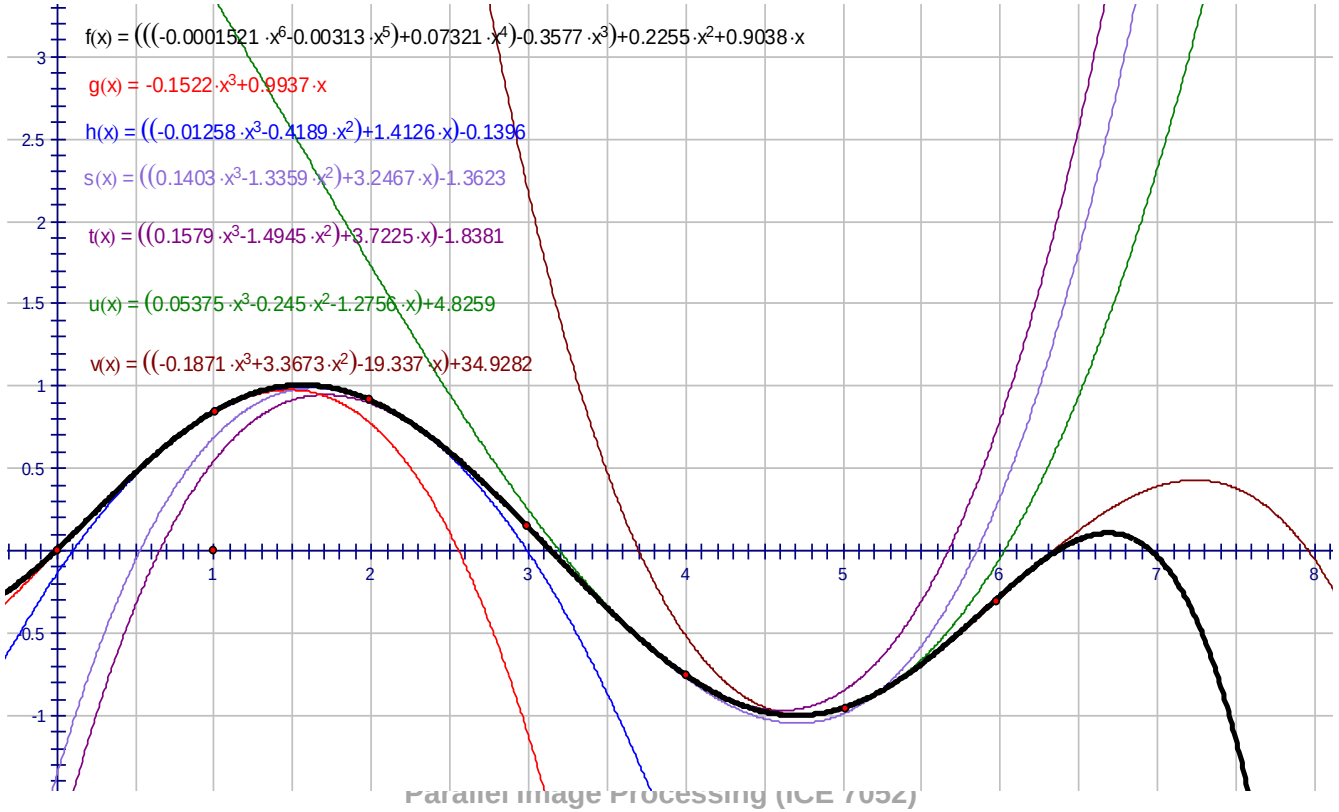
(3) Lagrange Interpolation

$$w_{lagra\_4} = \begin{cases} (1/2)|x|^3 - |x|^2 - (1/2)|x| + 1 & 0 \le |x| < 1 \\ -(1/6)|x|^3 + |x|^2 - (11/6)|x| + 1 & 1 \le |x| < 2 \\ 0 & elsewhere \end{cases}$$



(4) B-spline

$$w_{b\_spline} = \begin{cases} (1/2)|x|^3 - |x|^2 + 2/3 & 0 \le |x| < 1 \\ (-1/6)|x|^3 + |x|^2 - 2|x| + 4/3 & 1 \le |x| < 2 \\ 0 & elsewhere \end{cases}$$



Memory for input and out image

10	20	30
20	40	100
50	20	84

<Original input image>

Insert 2 pixels

horizontal

10	13	17	20	23	27	30
20	27	33	40	60	80	100
50	40	30	20	41	63	84

<Memory space for Interpolation result>

Memory for input and out image

vertical

10	20	30
20	40	100
50	20	84

Insert 2 pixels

10	13	17	20	23	27	30
13	18	22	27	35	45	53
17	22	28	33	48	62	77
20	27	33	40	60	80	100
30	31	32	33	54	74	95
40	36	31	27	47	69	89
50	40	30	20	41	63	84

<Original input image>

<Memory space for Interpolation result>

Programming steps

- Load image using OpenCV
- Decide the **number of pixels** for interpolation
  - Assumption : distance between pixels is the same
- Allocate the memory for output image
- Calculate the **weighting function** using the equation
  - Bileanaer , Bi-cubic, Lagrange, B-Spline
- Complete the interpolation function
  - **Function prototype**
    - Interp_omp(unsigned char * src, , int h, int w,char * wf , float * w, int x, int y, **unsigned char * output**)

Experimental results format

- Serial source code (given)
- **OpenMP using *section***
- Compare the processing time

Image	Size	# of Thread	nx /ny	Serial (*msec*)	OpenMP (*msec*)
Lena	512 × 512	4	1/1		
			2/2		
			3/3		
…	…	…			

References

[1]Rohit Chandra, Leonardo Dagum etc, “Parallel Programming in OpenMP,” Morgan Kaufmann Publichers, 2001

[2] https://computing.llnl.gov/tutorials/openMP/

[3] www.openmp.org